Self-service kiosk control program

Field of the invention

[01] The present invention relates to the field of web-enabled self-service kiosks such as automated teller machines, ticketing machines, vending machines, web kiosks and internet enabled interactive television set-top boxes with user interface peripherals.

Background

[02] Within this specification, self-service kiosks are networked computing devices designed for use by multiple users to carry out self-contained operations. Examples of kiosks are automatic teller machines (ATMs), public internet access terminals, credit-card operated and/or prepaid ticket collection terminals, networked ticketing machines and computerised vending machines as these are designed for use by multiple users and each user's interaction with the kiosk is a separate self-contained session. Home computers are not kiosks because users make open-ended changes to such computers; however, they can emulate kiosks and personal computers can also control kiosks. The present invention can also be applied to interactive televisions having the facilities to emulate kiosks. These facilities would include user identifying means such as a smartcard reader.

[03] "Web wraps" are known products which enable a kiosk to be limited to displaying only preset web pages. For example, a bookshop might provide a kiosk for accessing its on-line catalogue across the internet but block access to internet pages other than those within its own site. A web wrap has defined a subset of the internet and allows access to that subset.

[04] There is described below and claimed in a separate invention filed on the same day a method and software for carrying out customisations to kiosk function, including altering downloaded web pages and providing complex alternative or additional functions taking place concomitantly with web browsing. It is an aim of the present invention to enable customisations to kiosk function to be readily associated with subsets of the internet.

[05] It is a further aim of the present invention to make it easier to provide multiple different customised environments through a self-service kiosk. This will include provision of customised branding. Different customised environments can be associated with different subsets of the internet making it easy to provide multiple apparently different browsing environments through a single self-service kiosk.

Brief description of the invention

[06] According to a first aspect of the present invention there is provided a self-service kiosk control program comprising:

> a browser module for retrieving and displaying hyperlinked documents;
> at least one customisation activating module for activating customisations to default kiosk function, said customisations being associated with identified documents retrieved by the browser module; and
> the at least one customisation activating module being responsive to customisation information comprising information identifying one or more documents and information identifying at least one customisation associated with the identified documents.

[07] The invention therefore allows particular customisations to be associated with particular documents. Preferably, the identified documents are web pages identified by their URL and so different customisations can be applied to different sections of the internet. This allows customised presentation and branding and also customised transactions and complex operations to be performed for different sections of the internet. This makes is easier to prepare customised self-service kiosks and to enable a self-service kiosk to deal with web pages in a different fashion to when the same web page is displayed on a conventional computer.

[08] Preferably, the information identifying at least one customisation comprises a pointer to at least one customising software module for implementing said customisation and the customisation activating module activates said at least one customising software module whilst identified documents are being browsed by the browser module.

[09] The customisation information may comprise at least one rule or a reference to at least one rule and the identifier dependent module implements said at least one rule.

[10] The customisation information preferably specifies one or more conditional checks to be satisfied as a condition for performing said customisation. More preferably, the customisation information specifies one or more conditional checks to be satisfied as a condition for retrieving and/or displaying the identified documents. The customisation information may specify a check that a hardware device is operable. The customisation information may specify a check that an identified document is retrievable.

[11] Preferably, the customisation information identifies a document to be retrieved and displayed by the browser module when the browser first retrieves identified documents.

[12] Preferably also, the customisation information identifies a document to be displayed by the browser module when the user stops browsing documents identified by the customisation data.

[13] Typically, the document to be retrieved and displayed comprises executable instructions and/or script and wherein execution of said instructions and/or script causes the kiosk to perform specific functions relating to the end of a session.

[14] Preferably, default customisation information specifies default functionality of the kiosk and the identifier dependent module implements customisation specified by the default customisation information on initialisation or after each self-service session or responsive to an error or responsive to a set time having elapsed.

[15] The self-service kiosk control program is preferably adapted to limit a user browsing capability in accordance with customisation information. Customisation information typically includes a status parameter, wherein the customisation

activation module activates a customisation specified in said customisation information responsive to said status parameter.

[16] The identifier dependent module is preferably adapted to detect when a user browses to an identified document specified in customisation information and to implement said customisation responsive thereto.

[17] Preferably, customisation information specifies customisations to the visual display of identified documents. More preferably, customisation information specifies visual elements to be displayed in addition to the display of identified documents.

[18] Preferably, the identified documents are web pages.

[19] Typically, at least one of the identified documents can be used to implement a transaction and wherein a customisation is to perform said transactions in an order specified by the customisation information. Preferably, a customisation is the order in which the identified documents are presented to a user.

[20] In some embodiments, at least one of the identified documents can be used to implement a transaction wherein the customisation is to allow a specified transaction to take place only after at least one other identified transaction has taken place.

[21] Customisation information is preferably stored in a plurality of separate data structures, each data structure comprising information identifying at least one document and at least one customisation pertaining to said at least one document.

[22] According to a second aspect of the present invention there is provided a computer readable storage medium having at least one said data structure stored therein.

[23] According to a third aspect of the present invention there is provided a computer readable storage medium having stored therein the kiosk control program of the first aspect.

[24] According to a fourth aspect of the present invention there is provided a self-service kiosk control program comprising:

a browser module for retrieving and displaying web pages;

at least one customisation activating module for activating customisations to default kiosk function, said customisations being associated with identified web pages retrieved by the browser module;

the at least one customisation activating module being responsive to customisation information comprising information identifying one or more web pages and information identifying at least one customising module;

wherein the customisation activating module is adapted to activate the identified at least one customising module.

[25] According to a fifth aspect of the present invention there is provided a self-service kiosk control program comprising:

a browser module for retrieving and displaying web pages;

at least one customisation activating module for activating customisations to default kiosk function, said customisations being associated with identified web pages retrieved by the browser module;

the at least one customisation activating module being responsive to customisation information comprising information identifying one or more web pages and information identifying at least one customising module;

wherein the customisation activating module is adapted to activate the identified at least one customising module;

wherein said customisation information specifies one or more hardware devices which must be operable before retrieval of the identified web pages.

[26] According to a sixth aspect of the present invention there is provided a self-service kiosk control program comprising:

a browser module for retrieving and displaying web pages;

at least one customisation activating module for activating customisations to default kiosk function, said customisations being associated with identified web pages retrieved by the browser module;

the at least one customisation activating module being responsive to customisation information comprising information identifying one or more web pages and information identifying at least one customising module;

wherein the customisation activating module is adapted to activate the identified at least one customising module;

wherein said customisation information specifies one or more hardware devices which must be operable before retrieval of the identified web pages; and wherein said customisation information specifies the address of an initial web page to be retrieved when said customisations are activated.

[27] An example embodiment of the present invention will now be discussed with reference to the following Figures in which:-

[28] Figure 1 is a schematic view of a kiosk;

[29] Figure 2 is a schematic diagram of an architecture of a container software application;

[30] Figure 3 is a schematic diagram of a siteset;

[31] Figure 4 is a flow chart showing steps in completing and submitting an electronic form using information read from a card reader;

[32] Figure 5 is a schematic diagram of an architecture of an alternative container software application;

[33] Figure 6 is a schematic diagram of an alternative siteset;

[34] Figure 7 represents a screen display of an example user interface;

[35] Figure 8 is a schematic diagram of a preferred container software application architecture;

[36] Figure 9 is a flowchart of a method for printing documents from a self-service kiosk; and

[37] Figure 10 is a flowchart of an alternative method for printing documents from a self-service kiosk;

[38] Figure 11 represents a screen display of a compound transaction; and

[39] Figure 12 represents a screen display of a further example compound transaction.

Architecture

[40] Figure 1 illustrates a kiosk **1** having a display **2** driven by a computer **10** which has access to one or more servers **3** across a network **4**. Kiosks have user interfaces for interaction with users **5**. User interfaces may incorporate devices such as monitors, touch screens, keyboards, mice, cash dispensers, card reading devices, identification devices such as number pads for inputting a PIN number or cornea, iris or finger print readers.

[41] The system is particularly beneficial in the case where the computer **10** is connected to the internet and has the potential to access a large range of servers, but the owner of the kiosk wishes to customise the functionality provided by some web servers or restrict access to a subset of such servers under some or all circumstances.

[42] Figure 2 illustrates the architecture of software according to one aspect of the present invention. A container application **100** comprises one or more browser components **110** for accessing information from the internet and one or more site agent components **120** for providing customised functionality for particular uses of the kiosk.

[43] In the preferred embodiment, the container application **100** maintains an object model including all browser components **110**, site agent components **120** and also user interface **130** and transaction components **140** discussed below. Therefore, any component may communicate with any other through the object model.

[44] Typically, a browser component **110** has the capacity to access the internet and download information. It will typically also have the capacity to display web pages on screen **2**. Typically, the browser component **110** will be Internet Explorer® component from Microsoft®.

[45] Site agents adapt the way in which the computer **10** interacts with the web server **3** and user **5**. Different site agents are provided for different web applications.

[46] Site agent components **120** will typically be implemented in a scripting language and Microsoft® Script Host™ may be incorporated to enable a variety of scripting languages to be used. Alternatively, the site agent components **120** can be written in any standard programming language and precompiled site agent

components **120** can be loaded by the container application **100** on demand. Site agent components **120** can be preinstalled on a kiosk or can be downloaded from one or more remote servers **3** when required.

[47] Browser and site agent components may be separate components within the container application **100** or, alternatively, one may be contained within the other. For example, each site agent component **120** may have one or more browser components **110** therein or vice versa. Other configurations will be apparent to one skilled in the art.

Selection of site agents

[48] A site agent **120** is one way of implementing a document altering module. A simple document altering module which acts simply with reference to a series of rules specifying actions and the events which trigger those actions is discussed below.

[49] A site agent component **120** is triggered depending on one or more rules. For example, a kiosk **1** might have one site agent for use when accessing a particular server used by a bank and thereby providing functionality which is customised for the web application provided on that server. The same kiosk **1** may have a second site agent for use with an airline ticketing server and provide different functionality when that server is accessed.

[50] For example, Figure 3 illustrates a configuration where there is provided one or more data structures referred to as sitesets **200**. A siteset **200** is selected in response to user interaction with the kiosk. Typically, a siteset will have at least an identifier **201**, an initial address **202** in the form of a uniform resource locator (URL), one or more allowed address rules **203** and one or more site agent pointer **204** pointing to one or more site agent components **120** associated with that siteset. A siteset control **150** within the container application **100** activates customisations specified by sitesets **200**. The siteset control **150** provides methods such as a method to jump to a siteset, sneding a browser to the specified initial address **202** and starting up any specified site agent pointer **204**.

[51] A siteset **200** is typically selected in response to a trigger such as a user **5** selecting a menu item. A default siteset is activated initially, specifying an initial web page, initial site agent etc. The default siteset is reactivated after a timeout due to a user not carrying out any action for predetermined period of time.

[52] A browser component **110** is instructed to access the initial address **202**. Thereafter, access by the user to web pages is restricted to those allowed by the address rules **203**. An example address rule **203** would be to allow access to all web sites on a particular web server and no other sites. In this example, a rule **203** comprises a group of URLs, using wildcards to define a group in a single statement, for example, http://www.kal.com/* for all sites on www.kal.com. Use of wildcards enables types of document to be specified, for example http://*/*.exe defines all executable files and enables a rule to specify customisations depending on file type. For example, access to all executable files could be disallowed.

[53] A particular site agent indicated by the site agent pointer **204** is then started up. More than one site agent **120** may be activated at once and each may be associated with a browser component **110**. For example, multiple web sites may be displayed at the same time by multiple browser components **110** with each being associated with a different site agent component **120**.

Partitioning the internet

[54] The use of sitesets allows the internet to appear to a user of the kiosk as if it is partitioned. Particular functionality, determined by rules or site agents **120** specified in a siteset **200** is activated when a user accesses the subset of internet pages to which the siteset **200** pertains. This particular functionality can be accessed in response to a user requesting a particular siteset or may be activated automatically when a user browses to a location within the siteset. In another embodiment, the user is unable to leave the pages specified by the siteset until a specified procedure has been completed.

Required devices and required pages

[55] A siteset **200** may specify devices or web pages required to enable the siteset to be accessed. A site agent **120** may also require such a device or web page in order to function.

[56] Here, the kiosk checks whether a certain hardware device is functioning correctly or whether a certain web page is available. If a required device is unavailable or a required web page is unavailable the siteset or site agent is not activated. A siteset or site agent may specify alternative actions to be carried out should a required device or required web site not be available.

[57] As well as merely checking that a device or web site is available, the siteset or site agent may also check specified properties of the device or information returned in a web page to decide whether to allow the site set or site agent to be activated.

[58] An important function of site agents is to make modifications to the user interface presented to a user once a web page has been downloaded. Preferably, this is carried out using event driven programming. The container application **100** will ensure that the site agent is operative before the page is downloaded and the container application **100** has a module which activates one or more site agents **120**. If sitesets are used, the site agent will be loaded when the container application **100** moves to the new siteset. An alternative implementation might load the site agent when it detects navigation to a web page that the agent is associated with. In both cases, the browser signals the container application **100** when the page is downloaded, and the container application **100** passes control to the site agent prior to displaying the page. Alternatively the web page is displayed and the site agent is then called whereupon it amends the page and the browser updates the displayed page.

[59] Once a browser component **110** has downloaded a web page, a site agent **120** can then examine and, if required, alter the web page before it is displayed or take another predetermined action such as deleting executable script inappropriate to a kiosk. Examples of inappropriate script include defining stylesheets incompatible with the kiosk display or calls to methods requiring peripherals not found on a kiosk. Changes can be implemented by editing the script, or its document object model where relevant (see below). Changes to the visual presentation of a web site may include operations such as magnifying small details, deleting or customising logos,

altering style sheets and other document properties, removing components judged inappropriate for kiosk use such as the ability to search the entire internet from the downloaded page.

[60] For example, the browser component **110** can parse the web page into an object model according to an appropriate Document Object Model (DOM) such as the HTML or DHTML document object models described in standards available from Microsoft® or the World-Wide Web Consortium. The site agent **120** may then alter the object model before the web page is displayed.

[61] However, alterations to the visual presentation are not limited to merely altering the downloaded web page. In another embodiment a web page is downloaded and a site agent component causes a different web page to be displayed. Multiple different user interfaces may be layered and a particular interface selected. This enables a different web page to be displayed to that intended by the provider of a web site or provides a mechanism to readily provide displays in an alternative language.

[62] Figure 4 depicts an example of site agent operation. In this embodiment a web page is downloaded **225** from a server **3** by a first browser component **110**. In the depicted example the web page is a form requiring completion of credit card details to pay for a series of previously selected purchases.

[63] If the kiosk has direct access to a card reader, the site agent recognises **226** that alternative functionality is appropriate. For example, the form may require alphanumeric input of credit card authorisation details and the kiosk may have a credit card reader but no keyboard. Therefore, the best way for a user to enter their credit card details would be using the credit card reader provided as a peripheral on the kiosk. The site agent thus displays an alternative image saying "Swipe credit card" **227**. The alternative image may be stored locally in the kiosk or preferably be loaded from a remote web site. This alternative image is shown and the downloaded web page hidden. Alternatively, the site agent may cause a second browser component to acquire and display the alternative image on top of, and thereby hiding, the web page display region of the first browser component.

[64] The user's credit card details are then acquired **228** by the credit card reader whereupon the site agent **120** then causes these details to be submitted to the server **3**. The downloaded web page will typically have contained an electronic form to be completed by the user for submission to a server. The site agent **120** may electronically complete **229** this form (i.e. prepare an appropriate data record) using the information it has received from the credit card reader and cause the form to be submitted to the server **230**. The site agent then allows the next web page downloaded to be displayed in the normal manner **231**.

[65] Whereas altering the web page offers a wide latitude of modifications, the functionality of the site agent is preferably further enhanced by the capacity to 'call', or cause the execution of other software components and functions. Some such callable functions may activate and/or communicate with hardware components.

[66] Such hardware devices may be attached to the kiosk. For example, a site agent may use a kiosk peripheral such as a touch screen, a cash dispenser or a card reader to either cause particular hardware events to occur or to receive information (e.g. credit card details). This may be done directly or through software device driver components. Alternatively, the hardware device may be remotely operated such as a remote printer.

[67] A site agent may also use network access to carry out network functions, sending and requesting information.

[68] For example, a site agent might complete and submit an electronic form or activate (also known as 'drive') a particular web site, for example by making particular HTTP requests and extracting information from the resulting downloaded web pages. As well as just extracting information, remote web servers which provide functionality that depends on a user's previous requests can be brought to a particular state by performing said requests.

[69] Such requests can be made by means of a browser component which is not displayed on the screen.

[70] Examples include: (1) sending status to a monitoring application implemented as a web site (2) a combined shopping cart for a set of e-commerce sites, with the

agent filling in the details of each site as necessary. In this combined transaction scenario a site agent may drive remote web sites without displaying them to the user.

Triggering agent actions

[71] Site agents may function as linear applications or be event driven. Typically, a site agent will be notified by the container application whenever a navigation event occurs within a browser component associated with that site agent. However, events may be generated by other sources as well.

[72] The following are but several examples of events the site agent may be programmed to respond to:

- downloading of a page from a specified URL.
- downloading of a page containing a particular identifier. For example, a site agent may perform a function in response to a web page containing a particular meta-tag or password.

  a certain amount of time having passed – for example the time the user has spent using the kiosk.

  an event driven by another site agent, for example a site agent which calculates a user's bill for using the kiosk.
- a user triggered event, such as pressing a particular button.

[73] Site agents are preferably activatable; they can either be in activated or non-activated states and can be switched between the two by the container application, e.g. in response to user selection of a particular application from a menu.

Rule-based implementation

[74] Alternatively to using activatable site agents, the alteration of a document or carrying out of additional functionality may be carried out by a document altering module and/or function executing agent module which simply implements a series of rules without requiring activation.

[75] For example, such an agent module might scan each downloaded page and replace graphics in certain pages or increase font sizes which are below a size specified by a rule.

[76] Rules are event driven. Typically, a rule will be triggered by the container application whenever a navigation event occurs within a browser component associated with that site agent. Rules essentially consist of directives to be performed when particular activating conditions are met.

[77] Example activating conditions that might cause rules to perform directed functions include:

- downloading of a page from a specified URL.
- downloading of a page containing a particular identifier. For example, a rule may perform a function in response to a web page containing a particular meta-tag or password.

  a certain amount of time having passed – for example the time the user has spent using the kiosk.

  an event driven by another site agent, for example a site agent which calculates a user's bill for using the kiosk.
- a user triggered event, such as pressing a particular button.

[78] Rules will typically be implemented as pairs of activation conditions and directives. The activation conditions will encode a trigger event of the sort listed above, and the directive may specify any suitable functionality. Rules may be written as text and interpreted by the container application, or written in some suitable programming language and interpreted by the container application, or compiled to a suitable binary form for adding functionality to the container application. Rules can be preinstalled on a kiosk or can be downloaded from one or more remote servers when required.

[79] For convenience, rules may be grouped in rule sets. An implementation that uses sitesets might associate a rule set with each siteset. Rule sets could be downloaded separately from a server.

[80] An important, albeit not necessary, function of rules is to make modifications to the user interface presented to a user once a web page has been downloaded.

[81] For example, once a browser component has downloaded a web page, rules may comprise a directive to alter the web page before it is displayed. The browser component can parse the web page into an object model according to an appropriate Document Object Model (DOM) such as the HTML or DHTML document object models described in standards available from Microsoft® or the World-Wide Web Consortium. A directive may then alter the object model before the web page is displayed.

[82] Example changes to the visual presentation of a web site would be to magnify small details, delete or customise logos, alter style sheets and other document properties, remove components not judged appropriate for kiosk use such as the ability to search the entire internet from the downloaded page.

[83] However, alterations to the visual presentation are not limited to merely altering the downloaded web page. In another embodiment a web page is downloaded and a rule activated by download may direct that a different web site is displayed. Multiple different user interfaces may be layered and a particular interface selected.

[84] A rule may also direct that alternative functionality to that provided on the web page is appropriate. For example, the kiosk may have a credit card reader but no keyboard. Therefore, the best way for a user to enter their credit card details would be using the credit card reader provided as a peripheral on the kiosk. Example card readers include the KD Electronics CIM4900, the Mag-Tek MT215 and the Omron 3S4YR-MVFW. A rule may cause the container application to display an alternative image to show the user, for example a different HTML script stored within the computer or downloaded from another web site and saying "Swipe credit card" **227**. This alternative image is shown and the downloaded web page hidden. For example, the rule may instruct the container application to instruct a second browser component to acquire and display the alternative image on top of, and thereby hiding, the web page display region of the first browser component. Further rules can then direct the container application to input the data from the credit card in specific fields in the web form.

Combining site agents and rules.

[85] Rules need an interpreter to perform the actions they specify. In one embodiment this interpreter is the container application **100**. Another possible embodiment is for the container application **100** to be extended with site agents as described above, and for one or more of those site agents to use rules to configure their behaviour. Other possibilities include making the browser components or the transaction components use rules to customise their behaviour. Yet more possibilities will be apparent to those skilled in the art.

[86] Rules may also control how the container application uses network access to carry out network functions, sending and requesting information.

[87] For example, one group of rules might direct the completion and submission of an electronic form or drive a particular web site, for example by making particular HTTP requests and extracting information from the resulting downloaded web pages in response to the specified triggers. As well as just extracting information, remote web servers which provide functionality that depends on a user's previous requests can be driven by performing said requests.

[88] Such requests can be made by means of a browser component which is not displayed on the screen.

Layouts and user interface components

[89] Figure 5 illustrates another, embodiment of the present invention in which there are further provided one or more user interface components **130**. User interface components **130** are typically implemented as ActiveX® controls, JavaBeans® or the like, or they may use functionality built-in to the container application, such as specifying an area of screen to act as a button (or generate an event when touched by the user). The user interface component may also interface with various services provided by the operating system.

[90] User interface components **130** function to provide visual images and information to users when required. Properties, such as whether or not they are visible at a particular time, can be altered by site agents **120**, rule-based agents or

other software components. User interface components **130** can provide events and communicate with other components in response to user interaction. For example, a user interface component may be a banner for a particular service. When the banner is selected by a user, information stored in a siteset may be used to access a particular service.

[91] User interface components may be activated responsive to site agents, a rule-based agent module or instructions within web pages.

[92] Figure 6 shows an improved siteset **300**. Said siteset is as previously described with the addition of layout information **310** describing how displayed components should be laid out, including the number, locations and size of browser and user interface components. Thus one function of the siteset is to dictate the visual appearance of a downloaded web page.

[93] Figure 7 illustrates an example screen view **400**. A browser component provides a visible web content windows **401**. User interface components provide for example, a banner **402** which, if selected, provides a hyperlink to a web site. User interface components may include all types of information delivered through the internet such as streamed video or audio or, as provided in this example, an animation **403**. User interface components **404**, **405**, **406** may link to specific application by triggering sitesets **200** as described elsewhere herein.

Transaction processing components

[94] In Figure 8 a preferred embodiment is shown in which there are provided an extensible library of transaction processing components **140**. Transaction processing components **140** provide methods for completing hardware dependent tasks; for example, a single transaction processing component **140** might control ejecting a credit card, identifying a user from a fingerprint or carrying out complex protocols such as seeking authorisation for a financial transaction.

[95] Transaction processing components **140** can be accessed either directly from web page script executed within a browser component **110** or from other components. Transaction processing components **140** implement access to and interaction with hardware devices. In one embodiment these are OPOS Controls as

described in the OPOS specification or Kalignite® controls as provided by Korala Associates Limited, Edinburgh, Scotland.

[96] A transaction processing component **140** may use browser components **110** to access particular web pages or provide input/output functionality. For example, to download and display a web page stating that cash is being counted.

Compound transactions

[97] An aspect of the present invention provides for a conditional completion of a group of transactions. Thus, the set of transactions, or a 'compound transaction', are interdependent and only come to completion if a certain set of conditions is met, e.g. the successful completion of each member of a predetermined set of transactions.

[98] Figure 11 illustrates a screen shot of a first example of the present invention being used to implement a compound transaction. In this system, a transaction can be defined which requires the user to carry out multiple network interactions before it can be authorised. For example, a user might select a book from a bookseller web site, browse to select a parcel carrier from a separate web site belonging to the parcel carrier and browse to authorise payment, for example from a banking web site. The transaction of purchasing the book can only take place once the parcel carrier and payment method have been established.

[99] In this first example, a compound transaction control module is provided. This module is preferably a site agent **110** but could equally well be part of the container application **100**. Firstly, multiple web pages **600, 601, 602** are displayed **650**. A button **610** is to be selected by a user to complete a transaction (here, the purchase of a book) but the web page **600** displaying the button **650** is amended by the compound transaction control module after download and before display to disable the button. Responsive to subsequent events, the compound transaction control module then detects that information has been received indicating individual parts of a transaction have been completed. In this example, it detects that browser module **601** has accessed a web page known by its address to be displayed only when a payment has been confirmed and that browser module **602** has returned a page known by a meta-tag contained within the page to indicate that a parcel

delivery company has been booked. Once completion of all the parts of the transaction are completed, web page **600** is amended again, activating the button **610**. The user may now complete the compound transaction by selecting the button **610** which triggers an HTTP request through conventional browser technology.

[100] In a second example, the compound transaction control module causes multiple web pages to be displayed in separate browser components. A button is disabled in one window as before and only reactivated when the compound transaction control module detects that each browser component has navigated to a preset page known to be displayed at the end of a particular transaction.

[101] A third example is illustrated in Figure 12. Here a web page is displayed in a browser window **650**, the function of which is to enable a user to make multiple bookings and payments, in this case a flight **651**, a hotel **652** and ordering currency **653**. The user is prompted to enter information concerning their credit card number **654**, name **655** and address **656.** The user then selects a button **657**, responsive to which the compound transaction control module instructs one or more browser modules to retrieve booking web pages relating to the flight, hotel and currency ordering from separate servers belonging to flight, hotel and currency companies. The compound transaction control module then completes form information on each of these web pages using the name, address and credit card information supplied and returns it to the relevant server, thereby booking a flight, hotel and currency. The compound transaction control module then monitors the web page returned by the flight, hotel and currency companies' servers, extracts information required to decide whether each of the bookings was processed successfully from the address or content of the returned web pages and then directs the main browser window **650** to one of two preset web addresses, depending on whether the three separate transactions complete correctly.

[102] In a fourth example, a compound transaction control module has a list of web addresses stored. A first web page is downloaded. Each link within that web page which links to a web address not in the list is disabled. The web page is then displayed. As a user navigates between pages, the compound transaction module monitors which addresses the user has visited and, for each page, again disables links to web addresses not in the list. When the compound transaction module

detects that each web address on the list has been visited, the compound transaction module navigates to another pre-set page. This enables a transaction to be defined requiring a user to visit each of a number of web pages specified in the list.

[103] Features from these four examples may be combined in different permutations. In general, a compound transaction control module on the computer can be initiated either by user selection of the particular transaction or in response to the user browsing to a particular known web address, or in response to information contained within a web site. A compound transaction control module may then cause other web sites which are related to other parts of the compound transaction to be displayed. This may be done within a different window of the same browser, by customising the browser to display multiple web sites next to each other or by activating multiple browser components or multiple windows within the same component.

The compound transaction control module may guide the user through a sequence of web pages from the various sites by navigating a single browser from page to page or by using one browser per web-site and controlling which browsers are displayed. Remote web sites may be driven by providing one browser per web-site to access the remote sites but not actually displaying any of the browsers; instead the display would be another browser showing one web page. The main browser would gather information from the user and the secondary browsers, and pass on the necessary subset of this information to each secondary browser to send to its web site. Driving of remote web sites is discussed further above.

[104] Typically, the compound transaction control module does not authorise a specified transaction until it has established that other preset tasks have been completed. A compound transaction control module might require tasks to be completed in a specified order. The completion of tasks is typically monitored by receiving web sites of specified address or containing specific information from a remote server; however, the compound transaction control module could equally detection completion of tasks by monitoring user actions, such as selecting buttons.

[105] The compound transaction control module typically activates a transaction by sending a request to a server or by activating a means for allowing the user to

activate the transaction, for example, by allowing them to select a link which was previously disallowed.

[106] A compound transaction control module can be a site agent **110**, specified in a siteset or could equally well be a part of the container application **100**. Accessing a particular siteset can trigger a particular compound transaction control module. Compound transaction control modules can also be downloaded on demand from servers.

Branding

[107] One use of the agent or rule-based document altering module is to provide a means for branding sections of the internet It is generally desirable to be able to apply uniform branding to a section of a companies web site. Furthermore, many web-sites on the internet provide links to material supplied by third parties and it is desirable to find a way to ensure that a person browsing those third party sites associates them with the referring party. This is current achieved in two main ways. Firstly, frames can be used to show a linked site within a broader frame from the referring parties web page. This works but clutters the screen up and users will often return to the links site in the future with no reminder of how they first reached it. Secondly, new web pages are written which have co-branding to ensure that user's know that they were referred to a particular service by another party.

[108] However, the document altering module of the present invention can be used to make alterations to sites without resorting to either of these approaches. For example, on browsing to a companies web site, a site agent can adapt each downloaded page to add some branding. This might consist in altering some graphics, HTML stylesheets, adding logos, watermarked pictures etc. Furthermore, it may continue to provide such branding even when the user leaves the companies web site. For example, a site agent can be activated at a first company's web site then, when a user links to a second company's web site it will continue to display the first company's branding thereby showing a link between the firms. Similarly activation of multiple site agents allows multiple branding to be provided at once. As well as being activated by site agents, branding elements can be implemented by the container application implementing specified rules.

[109] A branding site agent may act not just to alter the visual appearance of downloaded web pages but to carry out other functions, such as guiding a user through a web site or reporting usage of the web site as will be apparent from the discussion of site agent's above. Furthermore, it may place additional images around the screen or start up additional functionality. For example, it may open separate browsers or display user interface components.

[110] Such site agents may be downloaded from a particular web site or preinstalled in a computer. They may be limited to functioning only within web addresses specified in a siteset or they may persist for at least the remainder of the user's browsing session. A single branding site agent may be used with multiple sitesets.

Wrap-up

[111] Self-service kiosks need to have a strategy for performing final clean-up actions once a user has finished using the kiosk. These actions may differ depending on whether a user has complied with directions, whether there have been exceptional events or in the event that the user walks away.

[112] The kiosk container application **100** comprises a session end determining module adapted to establish when a user has finished using the system whereupon an HTTP request is made and a wrap-up web page is downloaded from the web. The page may instead be found locally. This may be carried out by a timer which establishes when a predetermined time has passed since the user last interacted with the kiosk, an infrared sensor to detect a user walking away or it may be called by other software components.

[113] In embodiments where there is provided a siteset **200**, said siteset may further comprise the URL of a wrap-up page. In embodiments where site agents are provided, a site agent may carry out the wrap-up function.

[114] A wrap-up page may simply contain a visual image which shows a selected message. Preferably, it also contain script which calls software components that are installed on the kiosk and implement end of session actions. For example, in embodiments with transaction components, a transaction component may be called which retracts an untaken paper print-out.

A wrap-up page may not actually be displayed; indeed, merely the fact that a particular HTTP request is made may be all this is required to wrap up a sessions with a remote server. For example, a connection to a payment authorising server may be finalised by sending a particular request and/or particular information which leads to an appropriate response by the server without any web page actually being downloaded or displayed.

[115] Where a wrap-up page is displayed, this may be done in a secondary browser displayed as well as or instead of a primary browser.

Additional alternatives

[116] As well as being used to display web pages, the present invention can be used in a kiosk for displaying any form of browsable hyperlinked document downloadable across a network. Typically, such hyperlinked documents are provided in the form of separate pages having individual addresses. Documents may comprise both data, particularly data specifying a visual display, and executable instructions or script.

[117] The preferred embodiment of hyperlinked information downloadable across a network is web pages in the form of HTML, XML or Dynamic HTML documents sent or received by means of the well-known HTTP protocol.

[118] The invention extends to computer programs in the form of source code, object code, code intermediate sources and object code (such as in a partially compiled form), or in any other form suitable for use in the implementation of the invention. Computer programs may be standalone applications, software components or plug-ins to other applications. Computer programs may be web pages. Computer programs may be embodied on a carrier, being any entity or device capable of carrying the computer program: for example, a storage medium such as ROM or RAM, optical recording media such as CD-ROM or magnetic recording media such as floppy discs. The carrier may be a transmissible carrier such as an electrical or optical signal conveyed by electrical or optical cable, or by radio or other means. Computer programs may be provided for download across the

internet from a server. Computer programs may also be embedded in an integrated circuit.

[119] Where the term credit card is used, it will be recognised by one skilled in the art that the same use may be made of a debit card, electronic cash card, electronic payment card, electronic wallet or personal identifier device.

[120] A reference to a web page being executed refers to it being rendered for display on a screen and/or executed as a script. Execution of a web page typically includes parsing it into a document object model.

[121] Different features discussed in this application may be combined in different combinations to those explicitly stated.

[122] Section headings are provided for the purposes of clarity only and are not limiting.

[123] Further modifications and alterations may be made within the scope of the invention herein disclosed.